Case study
Hotel Room Booking Management System

Imagine that you are assigned to develop a room booking system for WestShore Holiday Park (Website: https://www.westshoreholidaypark.co.nz/accommodation-napier.html) which has three types of accommodation: Motel Room, Studio, Self-Contained Units. All rooms have the following data: id, facility, and tariff.

- As for motel rooms, they include one more features: extra number of people.
- As for self-contained units, they include two more property: extra number of people, fully furnished (yes/no).
- As for studios, they include one more attribute: private kitchen (yes/no).

You will be developing this system and a simple application that allow us to do:

1. Show all rooms based on their type (room / studio / unit).
2. Search a room having a particular facility
3. Add a new room to the system.

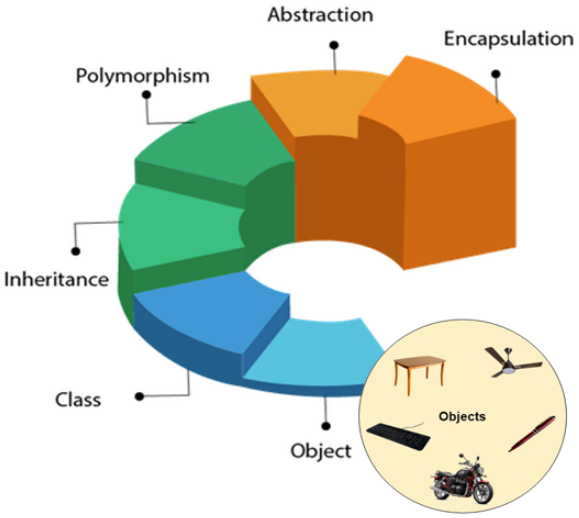
Step 1: Design the Class Diagram for this application

+ OOP Principles:

- **Objects** – Objects help us to **decompose** large systems & help us to **modularize** our system. **An object is the fundamental unit (building block) of a system**.
- **Class** – A class defines the **blueprint** (i.e. structure and functions) of an object.

- **Inheritance** – is a mechanism by which child classes **inherit the properties** of their parent classes.
- **Abstraction** – is a mechanism by which **implementation details are hidden** from user.
- **Encapsulation** – to bind data together and **protecting it from the outer world** is referred to as encapsulation.
- **Polymorphism** – is a mechanism by which functions or entities are able to **exist in different forms**.

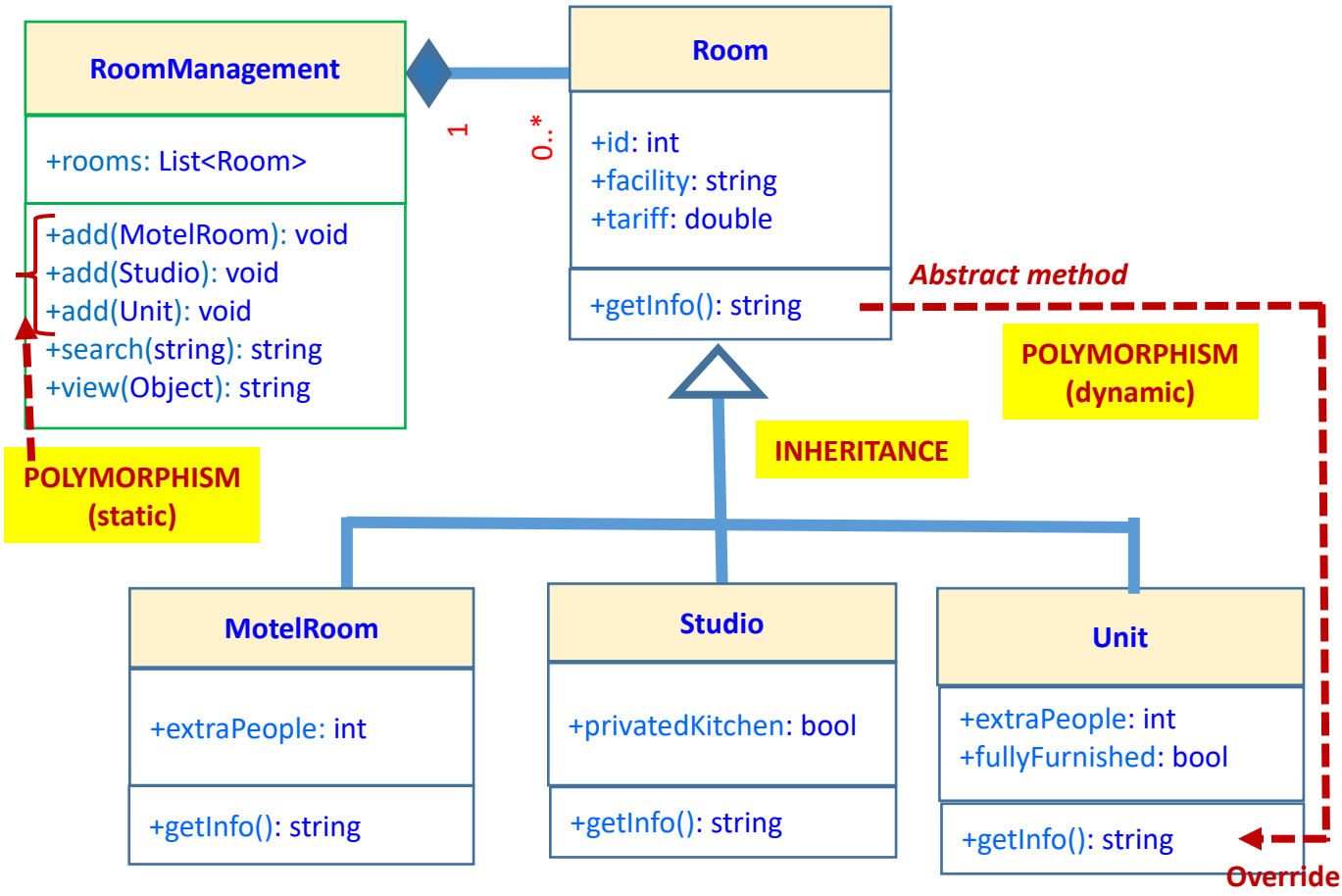
Open this online tutorial & have a look:
https://www.tutorialspoint.com/uml/uml_overview.htm



Four typical class relationship

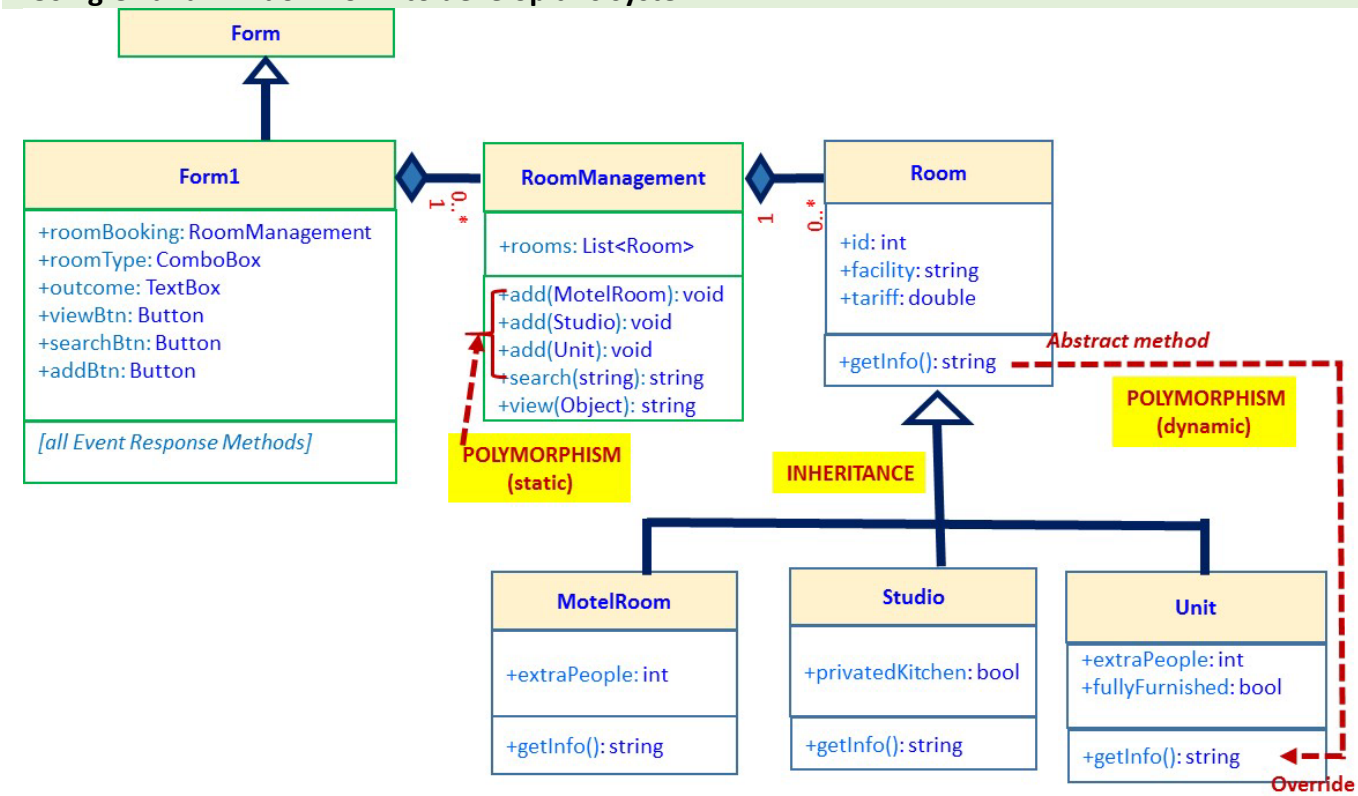
Relationship & symbol	Description
Inheritance ——▷	The relationship between super-class and sub-classes in **inheritance**.
Association ——→	A broad relationship: one object of this class owns an object of another class. **Example**: "Order" object has a "Customer" object as its property.
Aggregation ◇——	A subset of Association: when a class is formed as a collection of other class's objects. **Example**: "PlayList" class has a collection of "Song" objects.
Composition ◆——	A specific subset of Aggregation: where an object of a class cannot be existing outside of an object of another class. **Example**: An "Apartment" object is composed of some "Room" objects. A "Room" object cannot exist without an "Apartment" object; when an "Apartment" object is deleted, all associated "Room" objects are also deleted.

+ Class Diagram Outcome:



Step 2: Develop the system

+ Using C# and Window Form to develop this system:



+ Create a new Window Form App with the name *RoomBookingSystem*
+ Create a class called *Room* as a superclass:

- **Properties**: 3 properties with data types as mentioned in the class diagram.
  - "id" must be unique and auto-increment.
- **Constructor**: accepts 3 parameters (*facility, tarrif*);
- **Methods**: has a virtual method *getInfo()* that returns item information.

+ Inheritance & Polymorphism- create 3 sub-classes:

+ Create *MotelRoom* class

- This subclass **inherits** from the ***Room*** class;
- **Properties**: specific properties – extraPeople;
- **A constructor:** call the base class constructor
- **Methods**: override the getInfo() method which return the full information of this item.

+ Create *Unit* class

- This subclass **inherits** from the ***Room*** class;
- **Properties**: specific properties – extraPeople and fullyFurnished;
- **A constructor:** call the base class constructor
- **Methods**: override the toString() method which return the full information of this item.

+ Create *Studio* class

- This subclass **inherits** from the ***Room*** class;
- **Properties**: specific property – privateKitchen;
- **A constructor:** call the base class constructor
- **Methods**: override the getInfo() method which return the full information of this item.

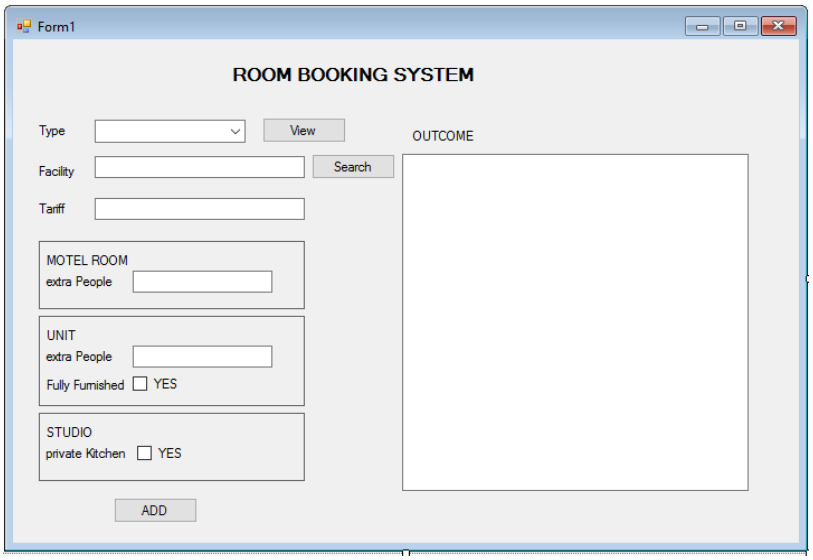
+ Create a class called *RoomManagement* and implement static polymorphism for *add()* method

Create a class called **RoomManagement**:

- **Properties**: the class has two attributes – *hotelName* (string and static) and a list of "*rooms*" that hold a collection of all items: *motel rooms*, *studios*, and *units*.
- **Constructor**: set the "*rooms*" list to 0 item.
- **Methods**: Add below methods:
  - **view(Object)** method: returns a string containing information of all items of the same type (Room/Unit/Studio) based on selected item in dropdown menu.
  - **search(string)** method that returns a string containing information of all items whose "facility" contains that keyword string [10 marks]
  - **3 add()** methods with many forms (static polymorphism): [10 marks]:
    - add(MotelRoom): add a new "motelroom" to "*room*" collection.
    - add(Studio): add a new "stuido" to "*rooms*" collection.
    - add(Unit): add a new "unit" to "*rooms*" collection.

+ GUI Test program

Add Controls to the Form to create a simple GUI:



- **Type**: is a ComboBox to create a dropdown menu with 3 items: MotelRoom, Studio, Unit.
- **3 panels** for entering data for 3 types of room
- **3 buttons**: "View", "Search", "Add"
- **1 Textbox (multiple lines)**: to display the outcome of each function.

+ Test Program

Write C# codes to implement the three below events (functions):

- **"View":** user select "type" (drop down menu), the application display all items of that type.
- **"Search"**: user enters key words, the application displays all items whose facility contain those key words;
- **"Add":** add a new item to the list of item.

Add the below items for testing:

- MotelRoom1: facility: "microwave, TV, heater"**. Tariff:** $120. **ExtraPeople**: 1
- MotelRoom2: facility: "microwave, TV, heater, hair dryer"**. Tariff:** $130. **ExtraPeople**: 1
- MotelRoom3: facility: "TV, free wifi, heater"**. Tariff:** $150. **ExtraPeople**: 2
- Studio: facility: "TV, heater, hair dryer, iron, radio"**. Tariff:** $170. **privateKichen**: true
- Self-contained Unit1**:** facility: "TV, heater, hair dryer, Iron"**. Tariff:** $170. **ExtraPeople**: 2. **fullyFurnished**: true